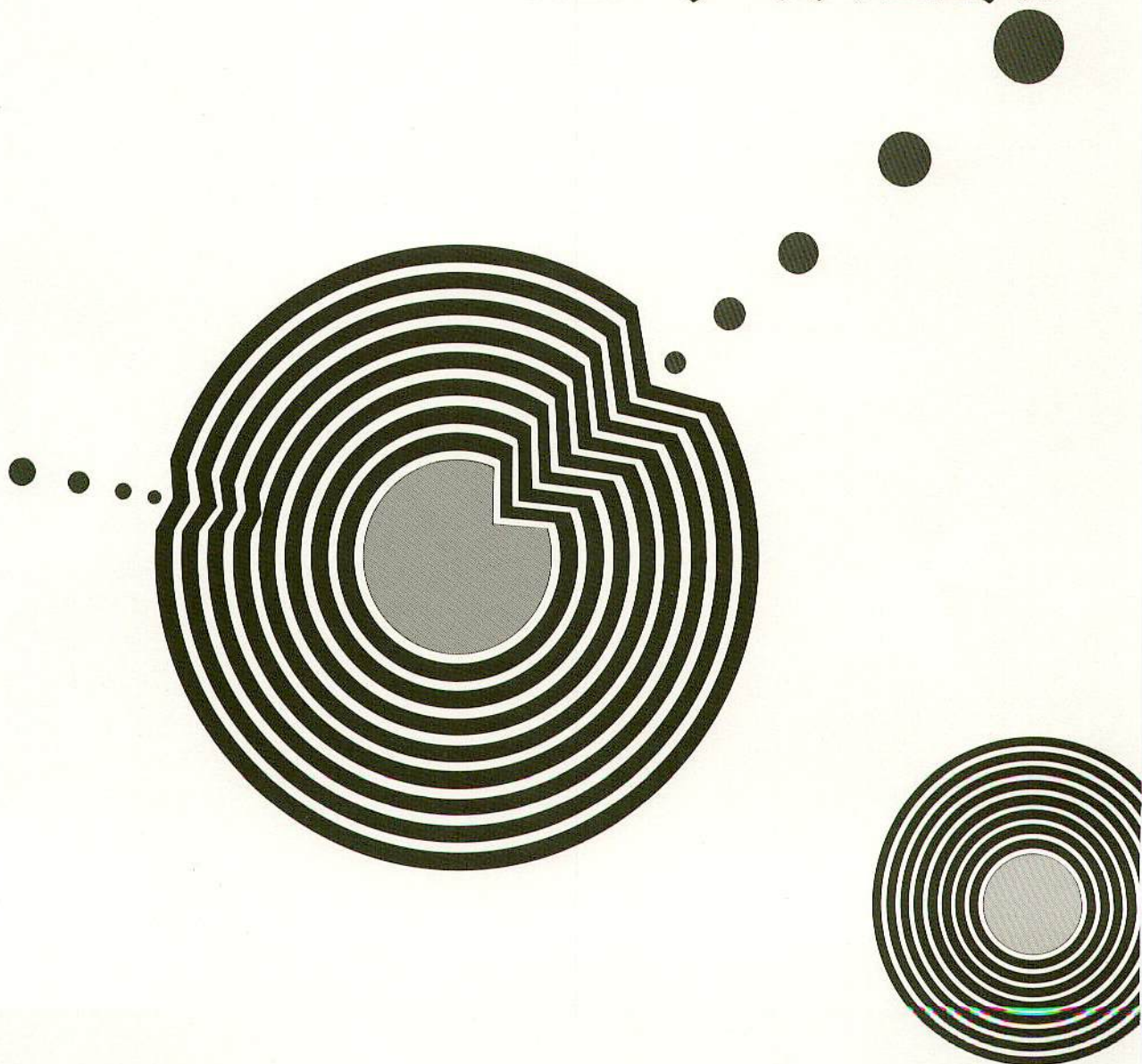
MSX 2 DOS2

日本語MSX-DOS2専用

MSX-S BUG2

エムエスエックス エス バグ2

ユーザーズマニュアル

ASCII
ASCII CORPORATION

はじめに

このたびは，MSX-S BUG2（エム・エス・エックス・エスバグ2）をお買い上げいただきまして，誠にありがとうございます．

MSX-S BUG2はMSX-DOS Ver2.xx（以下MSX-DOS2）に対応したシンボリック・デバッガです．MSX-DOS2により，階層ディレクトリ，RAMディスク等の使用が可能となりましたが，MSX-S BUG（以下MSX-S BUG1）はMSX-DOS2に対応していないため，デバッグ作業にこれらの機能を活用できませんでした．

MSX-S BUG2は，漢字モードの場合，ヘルプメッセージ（?コマンド），DUMPコマンド（Dコマンド）のASCIIキャラクタ出力を漢字で表示します．また，デバッガ本体をMSX-DOS2マッパーRAMに転送することによりユーザー使用領域を大きく広げることができるようになり，一層使い勝手のよい強力なシンボリック・デバッガとなりました．

なお，本ソフトウェアにはMSX-DOS2，スクリーンエディタ，MSX・M-80 Version 2.00（以下MSX・M-80），MSX・L-80 Version 2.00（以下MSX・L-80），LIB-80 Version 2.00，CREF-80 Version 2.00，MSX-C Ver.1.2は含まれていません．MSX-DOS2の機能を使ってC言語のプログラム開発を行うにはこれらのソフトウェアが必要となりますので，お持ちでない方は「日本語MSX-DOS2」，「MSX-C Ver.1.2」，「MSX-DOS2 TOOLS」をお買い求め下さい．また，アセンブリ言語を使ってプログラム開発を行うには，MSX-C以外の上記ソフトウェアが必要となりますので，お持ちでない方は「MSX-DOS2 TOOLS」をお買い求め下さい．

本ソフトウェアをご使用になる前に，添付の「ソフトウェア使用承諾契約書」をよくお読みいただき，ご確認のうえ，添付のユーザー登録カードにご記入し，弊社までご返送下さい．ご返送を持って承諾契約書にご同意いただいたものといたします．

この「ユーザー登録カード」をご返送いただけない場合には，弊社といたしましては所定のアフターサービスをいたしかねますので，よろしくご了承のほどお願い申し上げます．

本パッケージには以下のものが含まれています．

- MSX-S BUG2　システムディスク　1枚
  (3.5-1DDフロッピーディスク)
- MSX-S BUG2　ユーザーズマニュアル　1冊

ご注意

(1)このソフトウェアならびにマニュアルを賃貸業に使用することを禁じます．また，このソフトウェアやマニュアルの一部または全部を無断でコピーすることはできません．

(2)このソフトウェアは，個人使用以外の目的でコピーすることはできません．

(3)このマニュアルに記載されている事柄は，将来予告なく変更することがありますが，当社に登録されている方にはご案内をお送りします．

(4)製品の内容については万全を期しておりますが，製品の内容についてのご不審や，誤り，マニュアルの記載もれなど，お気付きのことがございましたら，マニュアルの巻末の「お問い合わせ」についての要領で下記の問い合わせ先へお送り下さい．

(5)このソフトウェアを運用した結果の影響については，(4)項にかかわらず，責任を負いかねますのでご了承下さい．

お問い合わせ先　〒107-24 東京都港区南青山6-11-1　スリーエフ南青山ビル
株式会社アスキー　ユーザーサポート係

TEL. 03-498-0299(祝祭日を除く月～金)
10：00～12：00，13：00～17：00

目次

本マニュアルについて

本マニュアルはMSX-S BUG2の各コマンドを十分に使いこなし，効率のよいデバッグ環境によりプログラムの開発効率を上げることを主眼に書かれています．したがって，デバッガを使うにあたっての基本的な事項については序章において簡略に述べるにとどめ，冊子のほとんどをMSX-S BUG2のコマンドの解説に割いています．

デバッグはアセンブラの知識はもちろん，ソフトウェア，ハードウェアを問わずおよそマイクロコンピュータに関するあらゆる知識を総合してかからなければならない作業であり，プログラム開発者にとってもっとも困難な仕事です．つまりデバッガを本当の意味で使いこなすには，本マニュアルの内容は最低限の操作方法に関する事項にすぎないということです．言い替えれば，本マニュアルによりデバッガの操作を修得し，実践の作業に用いることにより，マイクロコンピュータに対する理解が一層深まっていくことになるでしょう．

序　章

デバッグ作業

エラーなくコンパイル，アセンブル，リンクといったプログラム開発のための一連の作業を終了し，実行可能なプログラム（オブジェクト・プログラム）ができ上がったとしても，それは，ソース・プログラムのレベルで，文法上や意味上のエラーがなくなったということに過ぎません．プログラムが目的通り動作するかというと別問題であり，むしろほとんどの場合には，なんらかの隠れた誤りにより完全には動作しないものです．

このような場合，まずはソース・プログラムにかえって，プログラムの細部を再検討することによりなぜ正常に動作しないのかを探るのがデバッグの第一歩です．C言語など高級言語といわれる言語でソース・プログラムが書かれている場合，プログラムの実際の誤動作の状況を見ながらソースを検討することでかなりのバグを退治することが可能です．しかしながら，例えばC言語の場合でも関数のパラメータの型の不一致などによるバグなどは，ソース・プログラムのレベルでは発見の難しいものであり，さらにアセンブリ言語によるプログラムの場合などはCPUの各レジスタの働きの一つ一つにまで注意しなければならず，ソースを眺めているだけですべてのバグを捕らえるのは至難の技と言わなければなりません．

そこで，デバッグ作業を少しでも効率良く行うための様々な機能を備えたデバッガが必要になるわけです．

デバッガによるデバッグ作業の最大の特徴は，実行可能なオブジェクト・プログラムを直接メモリに読み込んで，その内容を目に見える形に表示したり，あるいは実際にプログラムの一部を実行させたりしながらバグの原因を発見していくという点です．これにより，ソース・プログラムのレベルでは発見の難しかったバグも素早く退治することが可能なわけです．

シンボリック・デバッグ

シンボリック・デバッグとは，その名のとおりプログラム中で定義されたシンボル名を用いてデバッグすることです．

例えば，アセンブラ・ソース・プログラムの中で

```
func1::
        ld      a,(de)
        cp      CR
          •
          •
          •
```

というようにパブリック・シンボル（他のモジュールでも参照可能なシンボル）としてラベルの定義されたルーチンがあるとします．このソースをアセンブルおよびリンクすることにより生成されたオブジェクト・プログラム中では func1 というシンボル名そのものは意味が無くなり，func1 というルーチンが実際に置かれたメモリ上のアドレス番地に置き換えられます．したがって，例えば func1 で示されたルーチンがメモリの 200H 番地から置かれた場合，

```
call     func1
```

という命令のオブジェクト・コードを逆アセンブルしても

```
call     0200
```

のように実際のアドレス番地でしか表すことができません．

これに対してシンボリック・デバッグでは，ソース・プログラム中で使われているシンボル名(各種シンボルやラベル）でそのアドレスの値を表わすことができます．上記の例の場合でも逆アセンブルの結果は，

```
call     func1
```

とより分かりやすい表示となります．さらに D コマンドや L コマンド（ともに後述）などアドレス番地をパラメータとして渡すコマンドでは

```
-d 200
-l 200
```

のような指定のかわりに

```
-d func1
-l func1
```

というふうに指定することができるわけです．

シンボリック・デバッグを可能にするためにはデバッガを起動しデバッグするプログラムを読み込むと同時にシンボル・テーブル・ファイルを読み込まなくてはなりません．シンボル・テーブル・ファイルはリンク時にリンカのオプション・スイッチの指定により生成することができます．図 0.1 にシンボル・テーブル・ファイルのつくり方の概要を示します．詳しくはリンカ（MSX・L-80）のマニュアル等をご覧下さい．

ソース・プログラム
リロケータブル・マクロアセンブラ
MSX・M-80
リロケータブル・
オブジェクト・ファイル
リンクローダ　MSX・L-80
L-80実行時に、/Xをつけることにより、コム・ファイル(XXX.COM)のかわりにヘキサ・ファイル(XXX.HEX)を生成することもできる。
実行可能・オブジェクト・ファイル
シンボル・テーブル・ファイル
L-80実行時に、/Yスイッチをつけることにより、シンボル・ファイル(XXX.SYM)が生成される。

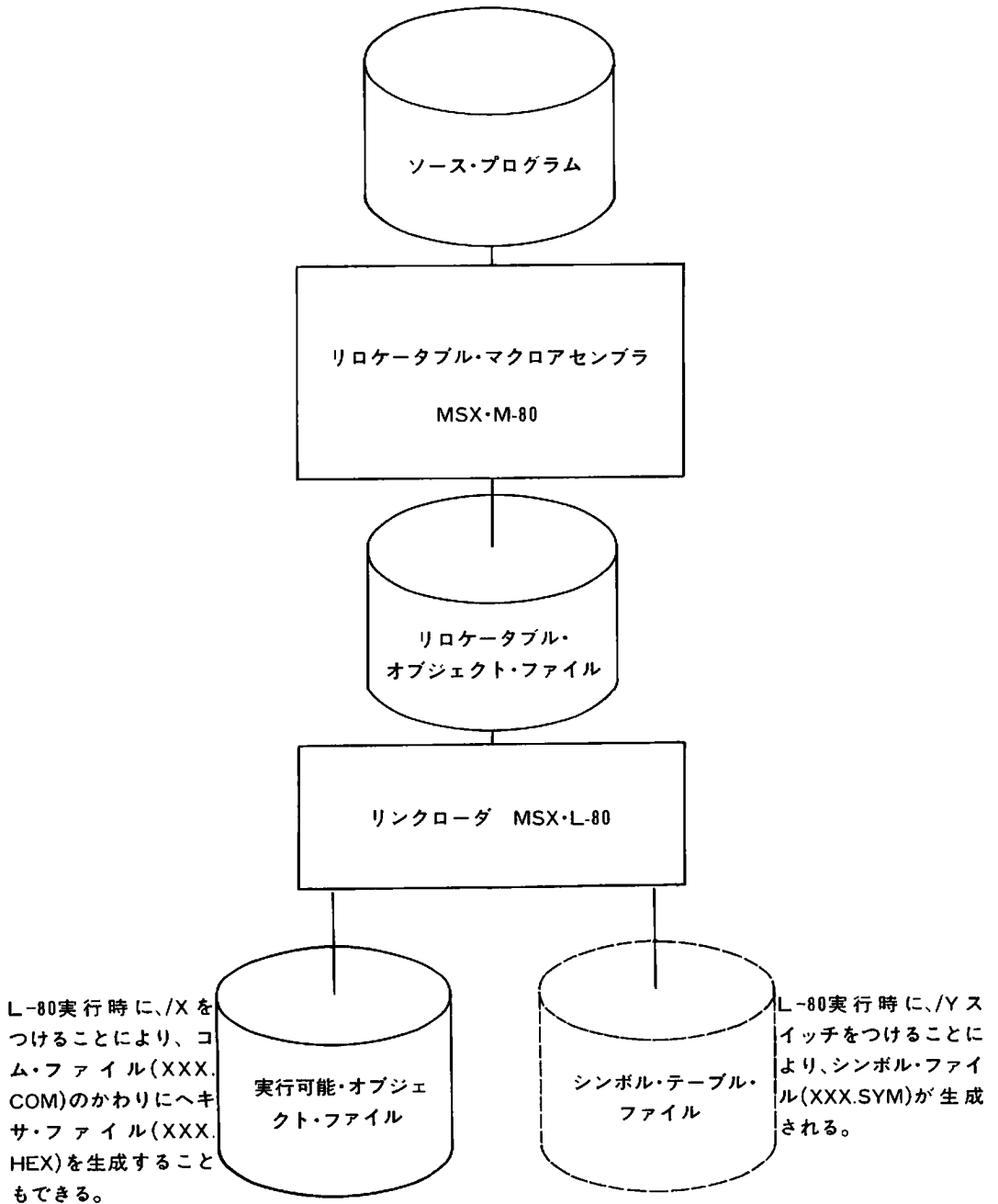



図0.1

デバッガの諸機能

デバッガには様々な機能があって，プログラムのバグ退治には欠かせないツールであることは前述のとおりです．それでは実際にデバッガにはどんな機能があるでしょうか．

ここでは，デバッガの標準的な機能を紹介します．

メモリ・ダンプ

メモリの内容を表示します．指定のメモリの内容を16進数とASCIIキャラクタで表示するのが一般的です．これによりプログラムが実際にメモリにロードされている状態が確認できます．

逆アセンブル

メモリの内容をオブジェクト・コード（マシン語命令）として解釈し，等価のアセンブラ・ニーモニックに置き換えて画面表示します．

メモリの内容を単純にアセンブラに置き換えるため，オブジェクト中のデータ領域を逆アセンブルしたり，逆アセンブルの開始アドレスを複数バイト命令の2バイト目以降に指定したりすると，不可解な結果となるので注意が必要です．

こうしたトラブルを防ぐためにもシンボリック・デバッグによりシンボル名でアドレスを指定することが重要になるわけです．

ユーザープログラムの実行

デバッガによるユーザープログラムのデバッグ中は，通常デバッガ自身が制御を握っているわけですが，一旦この制御をデバッグ中のユーザープログラムに渡し，これを実行することができます．もしデバッグ中のプログラムにシステムを暴走させてしまうようなバグがある場合は不用意にこれを行うとデバッガに二度と制御が戻らなくなりリセット・ボタンを押すはめになりかねません．しかし，デバッガからユーザープログラムを実行させるコマンド（以下，実行コマンドとよぶ）ではブレーク・ポイントを設定する事ができ，逆にプログラムがどのあたりで暴走するのかを特定するのにたいへん役立つのです．

ブレーク・ポイントの指定とは，ユーザープログラムの実行過程のどのアドレスでデバッガに制御を戻すかを，実行コマンドを行うときに予め指定しておくことです．プログラムが指定したブレーク・ポイントまで暴走せずに実行されれば再びデバッガに制御が戻ってくるわけですからブレーク・ポイントの位置を変えることによって暴走の原因のある範囲を狭めていけるわけです．

レジスタの内容の表示

CPU の各レジスタの内容が表示されます.

たとえば，前述の実行コマンドであるサブルーチンをコールした直後にブレークした場合，その時点でレジスタの内容を見ることができれば，いま実行されたサブルーチンが予想されたとおりの値を返してきたかどうかを知ることができるわけです.

トレース

メモリにロードされているユーザープログラムを1ステップずつ実行し，レジスタの内容をそのつど表示することができます．実行コマンドを使い，バグがあると思われるアドレスの直前でブレークしておき，1命令ずつトレースしていけば，もはやバグに肉薄したと言ってよいでしょう.

メモリの書き換え

バグの原因が判明して，なおかつプログラムの何バイトかを書き換えれば正常に動作することが判った場合，果して本当にそれでよいのかを確かめてみたい．そんなときにソース・プログラムを直してコンパイルやリンクをやり直すのはあまり効率的とは言えません．そこで，デバッガにはメモリにロードされたプログラムの任意の番地を直接変更する機能があります．これによりメモリの内容を変更し，前述の実行コマンドを使って正しく動作することを確認した後，メモリの内容をディスクにセーブしたり，あらためてソース・ファイルを更新すれば良いわけです.

アセンブル

前節で述べたメモリの書き換えは，メモリ中の定数部分の書き換えや，命令を1つか2つ，それもその命令のオブジェクト・コード（マシン語コード）がわかっている場合などに有効ですが，実際のデバッグ作業では場合によってはサブルーチンをまるまるひとつ変更したりすることもあります.

アセンブル機能は，こうした場合にユーザーの入力したアセンブラ・ニーモニックを等価のオブジェクト・コードにアセンブルし，任意のアドレスにそれを置いてくれるというものです.

メモリの書き換えやアセンブル機能によって一応動くようになったプログラムをディスクにセーブしても，それはあくまで一時的なテストの為のものと考え，ソース・ファイルに戻ってオブジェクトのレベルで変更した点をソースに反映させておくのが良いでしょう．そうすることによって，ソース・プログラムがバグ付きのままで放置されてしまったり，あとで他の人がソースをコンパイルしてもおなじオブジェクトができなかったりといったトラブルを防ぐことができるわけです.

以上，デバッガの機能の代表的なものを簡単に紹介しましたが，MSX-S BUG2 はこれらの標準的な機能はもちろん，メモリ・サーチ機能やコマンドのマクロ定義ができるなど多くの有用な機能を備えた強力なデバッガです．次章以下で解説する MSX-S BUG2 の各コマンドをマスターすれば他のデバッガを修得するのも容易な事となるでしょう．

第１章　MSX-S BUG2の概要

MSX-S BUG2 は MSX-DOS2 上で動作するシンボリック・デバッガです．MSX-S BUG2 は，通常のデバッガが備えている機能の他にメモリサーチ，マクロコマンドの使用，コマンドサマリーの表示などの強力な機能を持ち，さらに，漢字モード，階層ディレクトリなどの MSX-DOS2 の機能に対応していますので，アセンブラプログラム開発の大幅な効率化を図ることができます．

MSX-S BUG2 を起動するには MSX-DOS2 が必要ですのでご注意下さい．MSX-DOS Version 1.xx で立ち上げると以下のようなメッセージが表示され，MSX-DOS に戻ります．

```
A>sbug2
This program needs MSX-DOS2
```

なお，シンボル・テーブル・ファイルとしては MSX・L-80 が出力するもの，またはそれと同一形式のものを読み込むことができます．

ディスケットの内容

MSX-S BUG2 のディスクには次のファイルが含まれています．

sbug2.com
sbug2m.com

ただし，この内容は今後のバージョンアップに伴い変更されることがあります．

MSX-S BUG2のメモリマップ

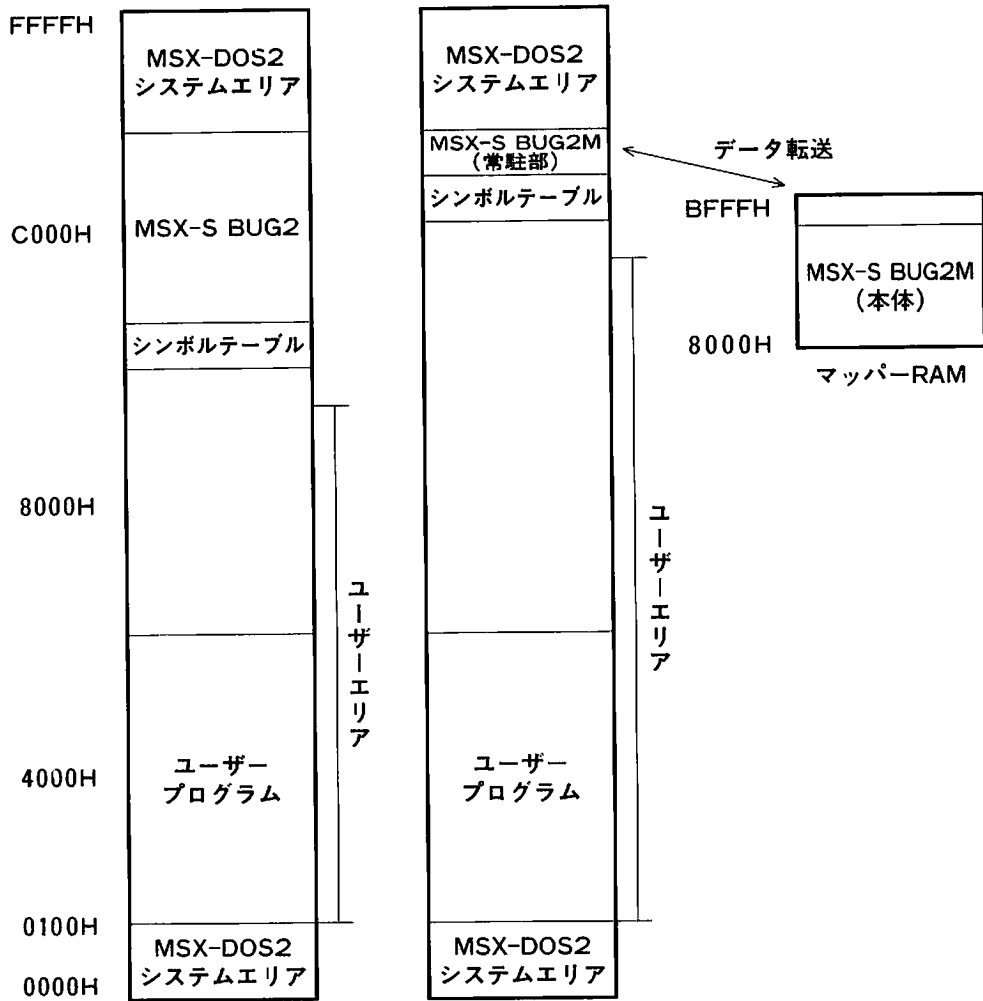



図1.1 sbug2.com　　図1.2 sbug2m.com

ディスクのバックアップ

お求めになったMSX-S BUG2のオリジナル・ディスクを破壊から保護するため，まずディスクのバックアップを行って下さい．すでにフォーマットされている別のディスクにMSX-DOS2のCOPYコマンドを用いて，マスターディスクに入っているファイルsbug2.com, sbug2m.comをコピーすればバックアップ完了です．なお，ディスクのフォーマット及び，COPYコマンドについては，MSX-DOS2のマニュアル等を参照して下さい．

第2章　MSX-S BUG1との相違点

MSX-S BUG2は，大部分の操作はMSX-S BUG1と変わりませんが，MSX-DOS2対応の機能に伴い，若干の機能追加及び変更が生じました．この章ではその変更点の解説をします．

2.1 sbug.com(MSX-S BUG1)とsbug2.com, sbug2m.com(MSX-S BUG2)の相違点

・階層ディレクトリ対応

ファイルを指定するとき，ディレクトリの指定をすることができます．ディレクトリの指定を省略した場合，カレントディレクトリを指定したことになります．

・MSX-S BUGのファイルI/O変更

デバッガの内部処理のファイルI/OはFCBを用いたものから，ファイルハンドルを用いたものになりました．デバッガの内部処理において，FCBをファイルI/Oとしてディレクトリの指定をすることは不可能なため変更となりましたが，これはデバッガ自身が行うI/Oであり，ユーザープログラムにとっては従来と変わりありません．

・コントロール-C, コントロール-STOPの機能変更

MSX-S BUG1ではコントロール-C, コントロール-STOPはデバッガ終了のためのコントロールキャラクタでしたが，MSX-S BUG2ではMSX-S BUG2のコマンド入力待ちに戻るためのものとなりました．

なお，デバッガ終了コマンド（Qコマンド）を追加しました．

・漢字モード対応

漢字モードであれば，DUMPコマンド(Dコマンド)のASCIIキャラクタ出力に正しい漢字が表示され，コマンドサマリーは日本語となります．漢字モードの画面構成の対応によりDUMPコマンドの画面構成も変更となりました．

DUMPコマンド（Dコマンド）の画面構成

画面モード	表示行数	表示データ数
ANKモード　(1行40字)	23行	8byte／行
〃　(1行80字)	23行	16byte／行
漢字モード　0	12行	8byte／行
〃　1	12行	16byte／行
〃　2	23行	8byte／行
〃　3	23行	16byte／行

・Q コマンド追加

MSX-S BUG2 より新しく加わったデバッガ終了コマンドです.

2.2 sbug.com(MSX-S BUG1)と sbug2m.com(MSX-S BUG2)の相違点

・デバッガ本体を MSX-DOS2 マッパー RAM に転送

デバッガ本体を MSX-DOS2 のマッパー RAM に転送して，ユーザー使用領域を大きく広げました. SBUG1，SBUG2 でデバッグできなかった大きなプログラムのデバッグに用意されたものです.

第3章　MSX-S BUG2の起動

MSX-DOS Version 2.xx 対応 MSX Symbolic Instruction Debugger (MSX-S BUG2) を使用することによって，ユーザープログラムのテスト，デバッグ，MSX-DOS2 のファイルのパッチなどを効率的に行うことができます．MSX-S BUG2 は sbug2.com, sbug2m.com の 2 つのデバッガ実行ファイルがあります．

sbug2.com は起動時にまず，メモリの 0100H 番地にロードされ，それから MSX-DOS2 オペレーティングシステムのユーザーエリアの最上位メモリ側の番地に転送されます．同時に MSX-DOS2 の BDOS コールのエントリアドレス (0005H 番地の飛び先アドレス) が MSX-S BUG2 の先頭番地に書き換えられ，ユーザーが使用できる空間の最上位メモリが変更されます．その結果 0100H 番地以降はユーザープログラムのために空けられ，MSX-S BUG2 自体は通常のユーザープログラムが直接関与しないシステム用の空間に移動して，テストのための環境が用意されたことになります．

sbug2m.com は起動する際，事前に MSX-DOS2 の RAMDISK コマンドで最低 16K バイト (1 セグメント) マッパー RAM を解放する必要があります．SBUG2M はここに自分自身を転送し，自分自身のあるセグメントをページ 2(8000H～BFFFH) に割当て，常にページ 2 のユーザー側と MSX-S BUG2 側との切り替え制御を行っています．sbug2m.com の起動の前に RAM ディスクを解放しておかなかった場合，

```
A>sbug2m⏎
Can not load to mapper ram
```

というメッセージが表示されて MSX-DOS2 に戻りますので，16K バイト (1 セグメント) 以上のマッパー RAM を解放して下さい．なお，RAM ディスクの設定方法は，「日本語 MSX-DOS2 リファレンスマニュアル」を参照して下さい．

sbug2.com と sbug2m.com の相違点はデバッガ本体の転送先のみで，操作，機能に違いはありません．

MSX-S BUG2 を起動するには，MSX-DOS2 のコマンドラインとして次のように入力して下さい．

1） SBUG2
    SBUG2M
2） SBUG2 <ファイル名>
    SBUG2M <ファイル名>
3） SBUG2 <ファイル名-1> <ファイル名-2>
    SBUG2M <ファイル名-1> <ファイル名-2>
4） SBUG2 /<コマンド-1> [；<コマンド-2>]･･･
    SBUG2M /<コマンド-1> [；<コマンド-2>]･･･

1)のように入力しますとMSX-S BUG2がメモリにロードされ，動作し始めます．起動後，ユーザープログラムをロードする場合はMSX-S BUG2のEコマンド，Rコマンドを利用します．これを簡単に行う方法として2)の書式があります．

2)のようにMSX-S BUG2の後ろにファイル名を指定すると，そのプログラムがMSX-S BUG2ロード後に自動的にロードされます．このときファイル名の拡張子が“.HEX”になっていると，ファイルはインテルフォーマットのヘキサファイルとしてバイナリに変換されてメモリにロードされます．それ以外の拡張子の場合にはアブソリュートバイナリファイルとして，0100H番地からロードされます．また拡張子を省略すると“.COM”ファイルをロードし，さらに，もしあれば同じ名前の“.SYM”ファイルをシンボルエリアにロードします．この場合，2つのファイルは同じディレクトリになければいけません．

3)のようにファイル名が2つ書かれ，しかも2番目のファイル名の拡張子が“.SYM”の場合には＜ファイル名-1＞のファイルが2)と同様にロードされた後，＜ファイル名-2＞のファイルがMSX-S BUG2のシンボルエリアにロードされます．拡張子が“.SYM”以外の場合には無視されます．

4)の書式を使いますとMSX-S BUG2起動後に実行すべきコマンドを“；”で区切って入力バッファが許すかぎりならべることができます．

例

漢字モード0での起動

1)
```
A>sbug2
MSX Symbolic Instruction Debugger 2.00
  (C)  BUG / ASCII  CORPORATION  1989
```

2)
```
A>sbug2 ¥test¥abc.com
MSX Symbolic Instruction Debugger 2.00
  (C)  BUG / ASCII  CORPORATION  1989

Next  = 3200
NextM = 3200
```

3)
```
A>sbug2 ¥test¥abc
MSX Symbolic Instruction Debugger 2.00
  (C)  BUG / ASCII  CORPORATION  1989

Next  = 3200
NextM = 3200
Symbol
```

4)
```
A>sbug2 /e¥test¥abc.com;r;d100 110;a100;ld a,(hl);.;s110;;;3
4;;.;q
MSX Symbolic Instruction Debugger 2.00
  (C)  BUG / ASCII  CORPORATION  1989
```

```
-e¥test¥abc.com
-r
Next  = 3200
NextM = 3200
-d100 110
0100  01 3E 2A C3 25 01 43 6F -省略-
0108  70 79 72 69 67 68 74 20
0110  28
-a100
0100  013E2A     LD    BC,2A3E
                 ld a,(hl)
0101  3E2A       LD    A,2A
                 .
-s110
0110 28 '('
0111 43 'C'
0112 29 ')'   34
0113 20 ' '
0114 31 '1'   .
-q

A>
```

sbug2m.com の起動は，起動メッセージを除き sbug2.com と同様です．sbug2m.com の起動メッセージは以下の通りです．

```
MSX Symbolic Instruction Debugger 1.00
            Mapper Version
  (C)  BUG / ASCII  CORPORATION  1989
```

第4章　MSX-S BUG2のパラメータ

4.1　デフォルト・レジスタ

MSX-S BUG2コマンドの一般形は以下の通りです．

<コマンド><パラメータ・リスト>[；<コマンド><パラメータ・リスト>]･･･

<コマンド>　　　　　　：コマンド名
<パラメータ・リスト>：0個以上のデータ／アドレスを表す式の並び

コマンドパラメータの数は各コマンドに固有ですが，コマンドによっては省略することが可能です．省略されたときに，代わりの値を提供するのがデフォルト・レジスタでA，D，G，I，L，OおよびSの各コマンドに独立に用意されています．MSX-S BUG2起動時のデフォルト・レジスタの初期値は，A，D，G，LおよびSの各コマンドが0100Hで，IとOが00Hとなっています．

たとえば，Dコマンド(メモリの内容を16進数とASCII文字で表示するコマンド)で，“D 100 200”と入力すると0100H番地から0200H番地までのメモリ内容を画面に表示しますが，このとき，“D ,200”と入力しても同様の動作をします．そして，Dコマンド用デフォルト・レジスタは0201Hに変更され，つぎに，“D ,300”と入力したときは，0201H番地から0300H番地までの内容が表示されます．

例

```
-i38⏎                 ⇨ 38Hポートから入力．Iコマンド用のデフォルト・レジスタの値は
FF    11111111           38Hに変更される．
-i⏎
FF    11111111        ⇨ 38Hポートから入力．
```

4.2　コントロール・キャラクタ

コマンドを入力するときに便利なように，各種のコントロール・キャラクタを使うことができます．コントロール・キャラクタは，キーボードのCTRLを押しながらアルファベットを押すことによって入力することができます．たとえばCTRLを押しながら“S”のキーを押すとコントロール-Sと呼ばれるコードが入力されます．MSX-S BUG2で許されるコントロールキャラクタの種類とその機能の一部を紹介します．

コントロール-H	最後の一文字を消し，カーソルをもどします．
コントロール-U/X	現在入力中の一行を無効にします．
コントロール-P	CRT に表示される文字と同じ文字をプリンタにも表示します．コントロール-P をプリンタが接続されていないときに押すと，CRT 上の出力がロックします．
コントロール-N	コントロール-P の動作を取り消します．
コントロール-C/STOP	コントロール-C，コントロール-STOP が押されたとき MSX-DOS2 のコマンド入力待ちに戻ります．(MSX-S BUG1 ではデバッガ終了のためのものでしたが，機能変更となりました．デバッガの終了は Q コマンドになります．) コントロール-C，コントロール-STOP キーで実行中のユーザープログラムをブレークすることができるのは，実行中のユーザープログラムが，コントロール-C，コントロール-STOP をチェックするファンクション(コントロール-C であれば文字 I/O ファンクション，コントロール-STOP であればステータスチェックを指定する文字ファンクションなど）を使用している場合に限ります． これは，MSX-S BUG2 が MSX-DOS2 の“アボート終了ルーチンの定義”ファンクションを使用して定義したアボートルーチンを用いているからです．ですからコントロール-C，コントロール-STOP キーを押すとアボートルーチンに飛んでくるファンクションを実行しているときに限りブレークできるのです． ユーザープログラムで“アボート終了ルーチン”ファンクションを行った場合，S BUG に戻るときに S BUG のアボートルーチンを定義するので，1 度 S BUG に戻ればユーザープログラムのアボートルーチンは無効となります． また，コントロール-C，コントロール-STOP でブレークした場合，PC はブレークが起こったファンクションコールの次の命令を指しています．
コントロール-S	表示を一時停止します．再び表示を開始したいときは，コントロール-P，N，C 以外の任意のキーを押して下さい．

また D コマンドなどを実行している最中にコントロール-S，P，N，C 以外のキー(コントロール･キャラクタでなくてもよい)を押すと，そのコマンドが中断されます．表示を中断したときには，デフォルトレジスタの値が最後に表示された値の次のアドレスに自動的にセットされます．

なお，1 行入力は MSX-DOS2 BDOS ファンクションコールを利用していますので，MSX-DOS2 のコマンドヒストリ機能を利用できます．その他のコントロール･キャラクタ及び特殊キーについては，「日本語 MSX-DOS2 リファレンスマニュアル」を参照して下さい．

4.3 式，定数，文字列

データ，アドレスを表す数として，2 進数，10 進数及び 16 進数を使うことができます．"0"～"9"，"A"～"F"，"a"～"f" からなる文字列，またはその直後に "H" をつけた文字列は 16 進数とみなされます．"0"～"9" の数字の並びの直後に "."（ピリオド）をつけた文字列は 10 進数，"0" または "1" の数字の並びの直後に "!" をつけた文字列は 2 進数，数値の後に何もつけていない文字列は 16 進数とみなされます．

また，数値は全て 16 ビットの符号なし整数として扱われ，桁あふれがある場合は 65536 の剰余がその値になります．

例

入力(基数)	値(10 進表現)
3 (16 進)	0003H (3)
5F (16 進)	005FH (95)
50. (10 進)	0032H (50)
10000 (16 進)	0000H (0)～桁あふれ
1024. (10 進)	0400H (1024)
1010! (2 進)	000AH (10)

数字以外に以下の特別なシンボルが用意されています．

#	現在のユーザースタックポインタの指し示す番地の内容．
$	現在のユーザープログラムカウンタの値．

"#"はサブルーチン内からもどり先のプログラムがどのようになっているかを見る時などに使われます．"$" は，これから実行するプログラムを見るときなどに使われます．

また，式の前に "^" を付けると間接参照を意味し，"^" の直後に（スペースを入れないで）書かれた式の値が指し示す番地の内容を値とします．

例

漢字モード 0

```
-d 100,12f⏎
0100  12 01 01 21 D9 30 20 01 -省略-
0108  F7 01 5A 2B ED B8 21 00
0110  CC 22 06 01 21 A6 ED 22
0118  5D DB 21 38 02 11 23 DD
0120  01 25 00 ED B0 21 5D 02
0128  22 4C F7 3E 00 32 4E F7
-h 100⏎
0100 256.
```

```
-h^100⏎
0112 274.
-h^^100⏎
0106 262.
-h^^^100⏎
0120 288.
```

ユーザープログラムカウンタ，ユーザースタックポインタの値は，“X”と入力することによって見ることができますが，“#” はユーザースタックポインタの値ではなく，スタックトップの値であるということに注意する必要があります．

“ ”（ダブルクォート）又は，‘ ’（シングルクォート）でかこまれた1文字又は2文字を定数として使うこともできます．この場合その文字列はアスキーコードとみなされて，数値に変換されます．

例

入力	値
"A"	0041H
'A'	0041H
"B1"	4231H
'3'	3320H

数字，変数，および文字定数は値を必要とする全ての場所で使うことができますが，さらに演算子と組み合せて式を書くことも可能です．演算子として許されるのは

*（乗算），¥（除算），&（AND）
-（単項演算子）
+，−，|（OR），^（XOR）
<，>，=，>=，<=，<>

で，演算の優先順位は，上記の表で上の方が強くなります．同じ優先順位の場合は式の左から右へ評価されます．優先順位を変更したい場合は“( )”のかわりに，“[ ]”を使います．これは“( )”を使うとザイログ・ニーモニックのアセンブラにおいて例えば

```
LD        HL,(100H)
```

と書いたとき，HL レジスタに直接 100H という値をロードしたいのか，0100H 番地の内容をロードしたいのか区別することができなくなるからです．

各演算子の意味は次の通りです．

*	16ビット符号なし整数の乗算
¥	16ビット符号なし整数の除算
&	16ビット符号なし整数のビットごとのAND
-(単項演算子)	"-a"は"0-a"に同じ
+	16ビット符号なし整数の加算
−	16ビット符号なし整数の減算
|	16ビット符号なし整数のビットごとのOR
^	16ビット符号なし整数のビットごとのExclusive OR

4.4 シンボルの参照

シンボル・テーブルがロードされている時には，式や定数の代わりにシンボルを用いることができます．

同じシンボルがいくつも登録されているときには，一番最後に登録されたシンボルの値が有効になりますのでご注意下さい．

シンボルとして許されているのは一文字目が，

@ . $? _ A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

のいずれかで，二文字目以降が上記の記号，英字または数字であるような文字列です．シンボルの最大長は7文字です．

例

漢字モード 0

```
-y⏎
2632 PRSYM      14F9 EXPRNN      275F MAKFCB      13A6 ASP
14E8 MSGOUT     14DC CRLF        0100 START       14F2 MSGEND
-h msgend-msgout⏎
000A 10.
-l msgout,msgend⏎
MSGOUT:
14E8  1A        LD    A,(DE)
14E9  13        INC   DE
14EA  FE00      CP    A,00
14EC  C8        RET   Z
14ED  CD4403    CALL  0344 [PUT]
14F0  18F6      JR    14E8 [MSGOUT]
MSGEND:
14F2  CDF914    CALL  14F9 [EXPRNN]
```

4.5 レジスタの参照

レジスタ名もシンボルと同様に式や定数の代わりに使用することができます．ただし，レジスタ名の前に“%”をつけてレジスタ名であることを明示しなくてはなりません．

レジスタ名として使用できるのは下記のとおりです．

F, F'	フラグ
A, A'	アキュムレータ
B, B'	B, B' レジスタ
C, C'	C, C' レジスタ
D, D'	D, D' レジスタ
E, E'	E, E' レジスタ
H, H'	H, H' レジスタ
L, L'	L, L' レジスタ
BC, BC'	BC, BC' レジスタペア
DE, DE'	DE, DE' レジスタペア
HL, HL'	HL, HL' レジスタペア
X, IX	IX レジスタ
Y, IY	IY レジスタ
S, SP	スタックポインタ
P, PC	プログラムカウンタ
I	インターラプト・ページ・レジスタ
R	リフレッシュレジスタ

例

漢字モード 0

```
-x⏎
MZ_H_ENC A =00 BC=5678 DE=3456 HL=1234
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=0100 P=0100  RST  38
-h %hl⏎
1234 4660.
-hbc⏎        ⇨レジスタ名よりも16進数の方が優先する.
00BC 188.
-h %bc⏎
5678 22136.
```

また，16進数と同じシンボルが定義されている場合には，シンボル名の方が優先します．

例

漢字モード 0

```
-y↵
2632 BC        14F9 EXPRNN     275F DE          13A6 ASP
14E8 MSGOUT
-hbc↵
2632 9778.
-dde,de+20↵
275F  3E 3F D3 E1 3D D3 34 CD -省略-
2767  00 02 C8 21 00 30 11 00
276F  40 3E 02 67 10 02 C3 00
2777  01 01 5A 2B ED B8 21 00
277F  58
```

このように16進数と同じシンボルが定義されている場合に，16進数の値を参照するには先頭に“0”をつけ，レジスタの値として使うには先頭に“%”をつけます.

例

```
-h bc↵
2632 9773.
-h %bc↵
5678 22136.
-h 0bc↵
00BC 188.
```

第5章 MSX-S BUG2のコマンド

MSX-S BUG2の各コマンドを具体的な例を挙げながら説明します．MSX-S BUG2のコマンドはCRT上に表示されるプロンプト文字“-”に続けて入力します．

なお，書式中の“_”は,<スペース>を表わします．

?　コマンド一覧の表示

機能

コマンド一覧を表示する．

書式

?

解説

MSX-S BUG2 の機能についてオンラインヘルプを提供します．“?” と入力することにより，いつでも MSX-S BUG2 のコマンド一覧を見ることができます．(付録 C Command summary 参照)

例

漢字モード 0

```
-?⏎
?             : 命令一覧の表示
A [番地]      : ニーモニック入力
C[N] [回数値] : CALL命令以外のトレース
D [開始番地] [終了番地]
              : メモリー内容の表示
E ファイル名 [ファイル名]
              : ファイル名のセット
F 開始番地 終了番地 データ
              : データによる初期化
G [開始番地] [/[番地[(回数値)]]...]
              : プログラムの実行
H 式 [式]     : 数値計算
Press any key to continue
```

A　アセンブル

機能

ユーザーが入力したニーモニックを機械語に変換してメモリ中に書き込む.

書式

1）A

2）A<式>

解説

<式>は，アセンブルの開始アドレスです．省略したときはデフォルトレジスタの値が使われます．MSX-S BUG2は入力されたアドレスの命令を逆アセンブルし，アドレス，コードそしてニーモニックを表示したあと入力待ちになります．ユーザーがニーモニックを入力すると，MSX-S BUG2はエラーのチェックを行い，正しい場合は機械語に変換してメモリに書き込みます．そして次のアドレスを計算し，再びアドレス，コード，ニーモニックを表示して入力待ちになります．もしニーモニックがまちがっていたときには，エラーを表示して再入力を要求します．“⏎”のみを入力すると，そのアドレスのコードは変更しないで次のアドレスに進みます．“.⏎”と入力するとMSX-S BUG2は再びコマンド入力待ちに戻ります．Z-80のニーモニックについては，Z-80アセンブラマニュアルを参照して下さい．

例

漢字モード 0

```
-a 120⏎
0120  5F        LD    E,A
                ⏎
0121  0E02      LD    C,03
                ld c,02⏎
0123  C30500    JP    0005
                .⏎
```

C　コール命令のルーチンをパスするトレース（レジスタ表示あり）

機能

コール命令のルーチンはパスしてトレースを行う．（レジスタ表示あり）

書式

1）C
2）C<式>

解説

書式1)では，プログラムを一命令だけトレースします．書式2)では，式の値がトレースする命令の数となります．従って“C1”と入力すれば，“C”と同じ意味になります．各命令をトレースするたびに，ユーザーレジスタの値をXコマンドとおなじ書式で表示します．Tコマンドとの違いは，CALL命令をトレースしようとした時に生じます．TコマンドではCALL命令の飛び先をさらにトレースしてゆきますが，CコマンドではCALL命令の飛び先まではトレースせず，そのサブルーチンから戻ってくるまでCPUはMSX-S BUG2の制御を離れます．従って，Cコマンドでトレースを行うときは，そのCALL命令で呼んでいるサブルーチンはすでにデバッグが終っていることが前提となります．

またTコマンドでトレースするプログラムはすべてRAM上に存在する必要がありますが，Cコマンドでは，CALL命令の先のサブルーチンは，ROM上にあってもかまいません．さらにそのサブルーチンはリアル・タイムで実行される点もTコマンドとは違っています．トレースを途中で中止したいときは，コンソールの任意のキー（コントロール-P，S，Nは除く）を押すと，MSX-S BUG2のコマンド入力待ちに戻り，次のコマンドを入力することができます．

なお，T，TN，C，CNコマンドでは，レジスタと一緒にそのアドレスのニーモニックが表示されますが，それはこれから実行する命令であって，すでに実行を終った命令でないことに注意が必要です．

<式>には式の他に数値を与えることもできますが，数値の後に何もつけていないもの，“H”のついたものは16進数とみなされます．10進数で表す場合は数値の後に“.”，2進数で表す場合は数値の後に“!”をつけます．式についての詳細は，第4章4.3式，定数，文字列を参照して下さい．

例

漢字モード 0

```
-c 2⏎
P _ _0   A =A1 BC=0000 DE=0000 HL=A168
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=00FE P=0108 CALL 010D

PZ_ _E   A =00 BC=0002 DE=0031 HL=0000
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=00FE P=010B  POP  HL
```

CN　コール命令のルーチンをパスするトレース（レジスタ表示なし）

機能

コール命令のルーチンをパスしてトレースを行う．（レジスタ表示なし）

書式

1）　CN

2）　CN<式>

解説

CN コマンドは，C コマンドが1命令トレースするたびにレジスタを表示することを除いてC コマンドと同じです．CN コマンドでは，最後の命令をトレースした後にだけレジスタを表示します．

例

漢字モード 0

```
-cn 2
M _H_ONC A =03 BC=0002 DE=0331 HL=0000
P _ _0   A'=00 B'=0000 D'=0000 HL=0000
X=0000 Y=0000 I=00
S=0100 P=011E  ADD  A,30
```

D　メモリのダンプ

機能

メモリの内容を16進数で表示する．

書式

1）　D

2）　D<開始アドレス>

3）　D<開始アドレス>_<終了アドレス>

4）　D　,<終了アドレス>

解説

表示される各行は，まずアドレスが，次に16個(ANKモード1行80字，漢字モード1，3の場合)又は8個(ANKモード1行40字，漢字モード0，2の場合)のデータが，そしてその値のASCII表現が並びます．

開始アドレスおよび終了アドレスとしては任意の式を書くことができます．書式1)では開始アドレスとしてDコマンド用のデフォルトレジスタが使われます．書式1)，書式2)では23行(ANKモード，漢字モード2，3)または12行(漢字モード0，1)表示すると自動的に表示が止まり，MSX-S BUG2のコマンド入力待ちに戻ります．また表示中にコントロール-Sを押すと表示が一時停止し，他の任意のキーを押すとそこで表示を中止します．デフォルトレジスタの値は次に表示すべきアドレスに変わります．書式3)では開始アドレスから終了アドレスまでを表示します．書式4)ではデフォルトレジスタの値を開始アドレスとして終了アドレスまでを表示します．

DUMPコマンド(Dコマンド)の画面構成

画面モード	表示行数	表示データ数
ANKモード (1行40字)	23行	8byte／行
〃　(1行80字)	23行	16byte／行
漢字モード 0	12行	8byte／行
〃　1	12行	16byte／行
〃　2	23行	8byte／行
〃　3	23行	16byte／行

例

ANK モード

```
-d 100 130
0100    44 49 52 0D 0A 0D 0A 5B 8B 40 94 5C 5D 20 20 83  DIR____[ぉ@え¥]  ◆
0110    66 83 42 83 58 83 4E 8F E3 82 CC 83 74 83 40 83  f◆B◆X◆Nって※ｱ◆t◆@◆
0120    43 83 8B 96 BC 82 F0 95 5C 8E A6 82 B7 82 E9 2E  C◆ぉｶｼ※みお¥ょう※ｷ※の.
0130    0D
```

（注意）　※は半角のクラブの代用で，平仮名，ダイヤは全て半角です．

漢字モード 0

```
-d 100 130
0100  44 49 52 0D 0A 0D 0A 5B DIR____[
0108  8B 40 94 5C 5D 20 20 83 機能]  デ
0110  66 83 42 83 58 83 4E 8F  ィスク上
0108  E3 82 CC 83 74 83 40 83  のファイ
0120  43 83 8B 96 BC 82 F0 95  ル名を表
0128  5C 8E A6 82 B7 82 E9 2E  示する.
0130  0D                      _
```

E　ファイル名の指定

機能

ファイル名を FCB にセットする.

書式

1）E<ファイル名>

2）E<ファイル名-1>_<ファイル名-2>

解説

このコマンドはファイル名を 2 つのデフォルト FCB(005CH～と 006CH～)に，コマンド入力行をデフォルトバッファ(0080H～)に書き込むために使います．書式 2)の場合，<ファイル名-1>が 1 番目の FCB に，<ファイル名-2>が 2 番目の FCB にセットされます．

しかし，MSX-S BUG2 は MSX-S BUG1 との互換性を保つため FCB にファイル名を書き込むだけで，実際にはファイルハンドルを用いてファイル I/O を行っています．

また，<ファイル名>のドライブ，ディレクトリを省略した場合，カレントドライブ，カレントディレクトリを指定したことになります．

例

1)
```
-e b:¥sample¥file.com↵
-e file.com↵
```

2)
```
-e ¥sample¥file↵
-e file↵
```

3)
```
-e ¥sample¥file.com ¥symbol¥file.sym↵
-e file.com file.sym↵
```

4)
```
-e ck,file,clib/s,crun/s,cend,file/n/y/c:xmain↵
```

このコマンドは一般に，R コマンドや W コマンドと組み合せて，ファイルの読み書きに使用します．

1)のように入力しますと，R コマンドであれば，ファイル名の拡張子が“.COM”であるためアブソリュートバイナリファイルとして 0100H 番地にロードされます．この拡張子が“.HEX”であるなら，ファイルはインテルフォーマットのヘキサファイルとしてバイナリに変換されてメモリにロードされます．W コマンドであれば，指定されたファイル名でメモリの一部をディスクに書き込みます．

2)のようにしますと，R コマンドであれば，拡張子を省略したファイル名であるため，“.

COM"ファイルをロードし，さらに，もしあれば同じ名前の".SYM"ファイルを MSX-S BUG2 のシンボルエリアにロードします．W コマンドであれば，拡張子を省略したファイル名でディスクに書き込みます．

3)は，R コマンドでのみ有効となります．".COM" ファイルと ".SYM" ファイルが異なるディレクトリに存在するという場合でも，このようにすればシンボルエリアにロードすることが可能です．2 番目のファイル名の拡張子が ".SYM" であるため "file.com" ファイルがロードされた後，"file.sym" ファイルが MSX-S BUG2 のシンボルエリアにロードされます．2 番目のファイル名の拡張子が ".SYM" 以外の場合には無視されます．

MSX-DOS2 のコマンド・プロセッサとおなじ効果がありますので，オプションを必要とするようなプログラムのデバッグにも利用できます．

4)は，180.com のオプションですが，180.com ロード後に 4)のように入力すれば，オプションをつけた状態の 180.com のデバッグが可能となります．

F　メモリの初期化

機能

メモリを初期化する.

書式

F<開始アドレス>_<終了アドレス>_<データ>

解説

開始アドレスから終了アドレスまでのメモリ・ブロックにデータを書き込みます. RAMのクリア, データの初期化などに使います. データとしては式が許されますが式の値は, 8bitの範囲（0〜255）に入っていなくてはなりません.

例

ANKモード

```
-f 100 17f 91⏎
-d 100 17f⏎
0100   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0110   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0120   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0130   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0140   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0150   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0160   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
0170   91 91 91 91 91 91 91 91 91 91 91 91 91 91 91 91   ああああああああああああああああ
```

漢字モード0

```
-f 100 17f 91⏎
-d 100 17f⏎
0100  91 91 91 91 91 91 91 91  荘荘荘荘
0108  91 91 91 91 91 91 91 91  荘荘荘荘
0110  91 91 91 91 91 91 91 91  荘荘荘荘
0118  91 91 91 91 91 91 91 91  荘荘荘荘
0120  91 91 91 91 91 91 91 91  荘荘荘荘
0128  91 91 91 91 91 91 91 91  荘荘荘荘
0130  91 91 91 91 91 91 91 91  荘荘荘荘
0138  91 91 91 91 91 91 91 91  荘荘荘荘
0140  91 91 91 91 91 91 91 91  荘荘荘荘
0148  91 91 91 91 91 91 91 91  荘荘荘荘
0150  91 91 91 91 91 91 91 91  荘荘荘荘
0158  91 91 91 91 91 91 91 91  荘荘荘荘
0160  91 91 91 91 91 91 91 91  荘荘荘荘
0168  91 91 91 91 91 91 91 91  荘荘荘荘
0170  91 91 91 91 91 91 91 91  荘荘荘荘
0178  91 91 91 91 91 91 91 91  荘荘荘荘
```

G　プログラムの実行

機能

プログラムを実行する.

書式

G<実行アドレス>／<ブレークポイント-1>･･･<ブレークポイント-3>

解説

パラメータはすべて省略することが可能です. Gコマンドを行うとCPUの各レジスタにユーザーレジスタの値が実際にセットされ, プログラムが実行されます. プログラムの実行は, 実行アドレスが指定された場合にはそのアドレスから, 省略された場合には現在のユーザープログラムカウンタの値から始められます. ブレークポイントの指定がある場合には, ユーザープログラムの実行中にいずれかのブレークポイントまで制御が移ってきたときに実行を中断し, コンソールに現在のレジスタの値が表示されます.

ブレークポイントに関しては次の点に注意する必要があります.

(1)ブレークポイントはRAM上のプログラムにのみセットすることができます. これは, ユーザプログラムからMSX-S BUG2に制御を戻すためのRST 28H命令をブレークポイントアドレスに書き込まなくてはならないためです. (そのアドレスのもともとの値はMSX-S BUG2の中のワークエリアに退避され, MSX-S BUG2に制御が戻ってきた際に復帰されます.)

(2)3個までのテンポラリ・ブレークポイントをセットすることができます. 4個以上ブレークポイントを入力するとエラーになります.

(3)G, T, Cコマンドが実行されると, 0028H番地にJP命令が書き込まれます. 従って, 0028H～002AH番地をユーザーが使うことはできません.

Gコマンドには, 更に各々のブレークポイントにパスカウント（通過回数）をセットする機能があります. パスカウントはブレークポイントの後ろに“( )”で囲んで指定します. パスカウントをセットすると, その回数だけこのブレークポイントを通過するまでMSX-S BUG2に制御が戻りません.

また, ブレークポイントを通るたびにキーボードが押されたかどうかがチェックされ, もしキーボードが押されていた場合はパスカウントが0にならなくてもブレークします. 設定したブレークポイント以外のアドレスでRST　28Hを実行すると, やはりMSX-S BUG2に制御が戻ります. これはユーザーが意識的にMSX-S BUG2に制御を戻したものと解釈されます. (例えば, ユーザープログラムが“JP　0”などにより正常終了した場合は, MSX-DOS2のコマンド入力待ちに戻ります.)

例

-g100/120(2) 123⏎

上記の場合，MSX-S BUG2 から 0100H 番地のプログラムに制御が移り，0120H 番地を2回目に通った時か 0123H 番地を始めて通った時に MSX-S BUG2 に制御が戻ります．

H　演算機能

機能

２つの数の和と差を１６進数および１０進数で表示する．

書式

1）　H<式>

2）　H<式１>_<式２>

解説

2つのパラメータとして任意の式を書くことが可能です．さらに2番目の式を省略することにより，簡単な2，10，16進電卓のように使うことができます．

なお，<式>についての詳細は，第4章4.3式，定数，文字列を参照して下さい．

例

```
-h 1251. 1↵
007E 126.  007C 124.

-h 400↵
0400 1024.

-h 100.+2*[101011!-12h]¥3&3eff |8000↵
8074 32884.
```

I　入出力ポートからの読み込み

機能

入出力ポートからデータを入力し16進数と2進数で表示する．

書式

1)　I

2)　I<ポートアドレス>

解説

ポートアドレスが省略されたときは，Iコマンド用のデフォルトレジスタが使われます．デフォルトレジスタの初期値は0000Hです．ポートアドレスを指定したときは，そのアドレスから入力されたデータがコンソールに表示され，デフォルトレジスタの値が指定したポートアドレスに変更されます．

例

```
-i 39↵
FF  11111111
-i 38↵
FF  11111111
```

IO　入出力ポートへの書き込み

機能

入出力ポートの内容を書き換える．

書式

1）IO

2）IO<ポートアドレス>

解説

I/Oポートに対するSコマンドです．メモリではなくI/Oポートに対してリード／ライトが行われることを除いては，Sコマンドと同じです．デフォルトレジスタはIコマンドと同じものが使用されるので注意が必要です．

例

```
-io 39⏎
0039 FF 11111111⏎ 3E    ⇨ 03EH をポート 0039H にアウト.
003A FF 11111111⏎ 20
003B FF 11111111⏎ .     ⇨ 020H をポート 003AH にアウト.
```

K　パーマネント・ブレークポイントの設定

機能

パーマネント・ブレークポイントを設定する.

書式

1) K

2) K<ブレークポイント-1>_<ブレークポイント-2>··<ブレークポイント-5>

解説

Gコマンドでセットされたブレークポイントはブレークが生じたときに無効になってしまいますが，パーマネント・ブレークポイントはいったんセットするとKXコマンドで消去されるまで有効です.

パーマネント・ブレークポイントは最大5個までセットすることが可能であり，現在セットされているブレークポイントを確認するには書式1)を使います.

書式2)はパーマネント・ブレークポイントをセットするときに使います.＜ブレークポイント＞はブレークアドレスとパスカウントを含み，アドレスの後ろに“( )”で囲んで式を書くとその値がパスカウントになります．パスカウントには省略すると自動的に1がセットされます．ユーザープログラムに制御がある場合，Gコマンド実行時にブレークポイントを通過し，しかもその通過回数がパスカウントと一致した場合にはユーザープログラムからMSX-S BUG2に制御が戻ります.

例

```
-k 110 120(2) 130(3) 140 150
-k
BP=   0110 :0001
BP=   0120 :0002
BP=   0130 :0003
BP=   0140 :0001
BP=   0150 :0001
```

KX　パーマネント・ブレークポイントの解除

機能

パーマネント・ブレークポイントを消去する．

書式

1）　KX

2）　KX<アドレス1>_<アドレス2>…

解説

このコマンドはKコマンドで設定したパーマネント・ブレークポイントを消去するのに使用します．書式1)では全てのパーマネント・ブレークポイントを消去し，書式2)では指定したアドレスにセットされているブレークポイントのみを消去します．

例

```
-k 110 120(2) 130(3) 140 150⏎
-k⏎
BP=   0110 :0001
BP=   0120 :0002
BP=   0130 :0003
BP=   0140 :0001
BP=   0150 :0001
-kx 130 140 150⏎
-k⏎
BP=   0110 :0001
BP=   0120 :0002
```

L　逆アセンブル

機能

アセンブルリストを表示する.

書式

1)　L

2)　L<開始アドレス>

3)　L<開始アドレス>_<終了アドレス>

4)　L　,<終了アドレス>

解説

メモリの内容をアセンブリ言語のニーモニックで表示するのに使用します.

書式 1)では開始アドレスとして，L コマンド用のデフォルトレジスタが使われます．デフォルトレジスタの初期値は 0100H です．書式 1)，書式 2)では開始アドレスから 16 行逆アセンブルして表示します．書式 3)では，開始アドレスから終了アドレスまでが逆アセンブルされます．書式 4)ではデフォルトレジスタの値から終了アドレスまで逆アセンブルされます.

表示終了後，デフォルトレジスタは終了アドレスの次にセットされます.

例

ANK モード

```
-l 100 111↵
0100    21 00 40      LD     HL,4000
0103    7E            LD     A,(HL)
0104    23            INC    HL
0105    B7            OR     A,A
0106    28 05         JR     Z,010D
0108    35            DEC    (HL)
0109    28 02         JR     Z,010D
010B    18 F6         JR     0103
010D    35            DEC    (HL)
010E    36 01         LD     (HL),01
0110    18 F1         JR     0103
```

漢字モード 0

```
-l 100 111↵
0100  210040   LD    HL,4000
0103  7E       LD    A,(HL)
0104  23       INC   HL
0105  B7       OR    A,A
0106  2805     JR    Z,010D
0108  35       DEC   (HL)
0109  2802     JR    Z,010D
010B  18F6     JR    0103
010D  35       DEC   (HL)
010E  3601     LD    (HL),01
0110  18F1     JR    0103
```

M　メモリ上のデータ転送

機能

メモリ上のデータを転送する.

書式

M＜ソース・アドレス＞_＜ソース・エンド＞_＜デスティネーション・アドレス＞

解説

このコマンドは＜ソース・アドレス＞から＜ソース・エンド＞までのメモリの内容を＜デスティネーション・アドレス＞に転送するものです.転送後のベリファイは行っていませんので場合によっては(メモリが存在していない場合や，テスト中のメモリ・カードで信頼性が低い場合など) V コマンドによってベリファイする必要があります．また，転送するブロックと転送されるブロックが重なっている場合でもスタートアドレスから順に転送するか，エンドアドレスから順に転送するかを MSX-S BUG2 が自動的に決定していますので，正しく転送が行われます．従って M コマンドを F コマンドのように特定のパターンでメモリ内容を初期化するコマンドとして使用することはできません.

例

漢字モード 0

```
-d 100 140⏎
0100  31 00 01 21 D9 30 11 59 -省略-
0108  F7 01 5A 2B ED B8 21 00
0110  CC 22 F5 DA 21 A6 ED 22
0118  5D DB 21 38 02 11 23 DD
0120  01 25 00 ED B0 21 5D 02
0128  22 4C F7 3E 00 32 4E F7
0130  97 32 04 00 3C 32 03 00
0138  CD 5E 02 CD 3C EC 0C 43
0140  50
-m 100 120 110⏎
-d 100 140⏎
0100  31 00 01 21 D9 30 11 59 -省略-
0108  F7 01 5A 2B ED B8 21 00
0110  31 00 01 21 D9 30 11 59
0118  F7 01 5A 2B ED B8 21 00
0120  CC 22 F5 DA 21 A6 ED 22
0128  5D DB 21 38 02 11 23 DD
0130  01 25 00 ED B0 21 5D 02
0138  22 4C F7 3E 00 32 4E F7
0140  50
```

N　メモリのサーチ

機能

メモリブロック内にある連続したデータをサーチする．

書式

N＜開始アドレス＞_＜終了アドレス＞_＜データ-1＞_＜データ-2＞・・・

解説

＜データ＞は値が0～255の範囲内の式です．また＜データ＞として“ ”（ダブル・クォート）または' '（シングル・クォート）で囲まれた任意長の文字列も許されます．この場合，気をつけなくてはならないのは＜データ＞として“AB”は許されるが，1+“AB”は許されないということです．“AB”はメモリ中に存在する41H，42Hという連続したパターンをサーチすることを意味し，1+“AB”は1+4142H＝4134Hという値となるためレンジエラー（許される値は0～255）となります．以下の入力は全くおなじ意味となります．

```
-n 100 200 "ABCDEF"↵
-n 100 200 41 42 43 44 45 46↵
-n 100 200 "A" 1+"A" [-1+"D"] "DE" 46↵
```

また，パターンがマッチすると，その先頭アドレスとそこから16又は8バイトのデータをコマンドとおなじ形式で表示します．

例

漢字モード 0

```
-n 100 200 "version"↵
0144  76 65 72 73 69 6F 6E 20 -省略-
014C  32 2E 32 20 2B 20 43 6F
-d 100 160↵
0100  31 00 01 21 D9 30 11 59 -省略-
0108  F7 01 5A 2B ED B8 21 00
0110  CC 22 F5 DA 21 A6 ED 22
0118  5D DB 21 38 02 11 23 DD
0120  01 25 00 ED B0 21 5D 02
0128  22 4C F7 3E 00 32 4E F7
0130  97 32 04 00 3C 32 03 00
0138  CD 5E 02 CD 3C EC 0C 43
0140  50 2F 4D 20 76 65 72 73
0148  69 6F 6E 20 32 2E 32 20
0150  2B 20 43 6F 70 79 72 69
0158  67 68 74 20 28 63 29 20
0160  31
```

O　入出力ポートへのデータ出力

機能

入出力ポートへデータ出力する．

書式

1）O<ポート>_<データ>

2）O<データ>

解説

I/O ポート<ポート>に<データ>を出力します．<ポート>を省略した場合には最後に指定された<ポート>に出力します．

注意

MSX システムではI/O ポートの 40H から FFH はシステム用に使用していますので，不用意にこのコマンドを実行するとなんらかの障害を生じることがあります．（MSX ハードウェアマニュアルを参照して下さい．）

例

```
-o 38 ff⏎
-o ff⏎
```

P　マクロ・コマンドの定義及び実行

機能

マクロ・コマンドを定義，実行する．

書式

1）　P<コマンド>［：<コマンド>］･･･

2）　P

解説

頻繁に使用するコマンド列を登録しておき，必要なときに“P”と入力することで，そのコマンド列を実行することができます．

書式 1）はコマンド列の登録です．コマンドとコマンドに付属するパラメータを，（マルチコマンドで区切りに使う“；”（セミコロン）ではなく）“：”（コロン）で区切って何個でも並べることができます．登録したコマンド列を実行するには，書式 2）のように“P”と入力するだけでよく，登録されているコマンドが左から順番に実行されます．

マクロ・コマンドは一度登録されると PX コマンドでクリアされるか再定義されるまでは，キャンセルされないので何回でも使うことができます．

例

漢字モード 0

```
-p d100:1⏎
-p⏎
-D100
0100  01 E4 32 C3 25 01 43 6F -省略-
0108  70 79 72 69 67 68 74 20
0110  28 43 29 20 31 39 38 32
0180  2C 20 42 20 55 20 47 20
0120  20 49 6E 63 2E C5 3A 07
0128  00 3D 90 F5 3D 21 00 02
0130  54 5D 09 E5 CD 46 01 E1
0138  2B D1 C1 1E 00 EB 09 2B
0140  EB ED B8 13 EB E9 E5 67
0148  78 B1 28 15 0B 7B E6 07
0150  20 05 E3 7E 23 E3 6F CB
0158  15 30 03 1A 84 12 13 18
-L
010E  74      LD   (HL),H
010F  2028    JR   NZ,0139
0111  43      LD   B,E
0112  29      ADD  HL,HL
0113  2031    JR   NZ,0146
0115  39      ADD  HL,SP
0116  3832    JR   C,014A
0118  2C      INC  L
0119  2042    JR   NZ,015D
011B  2055    JR   NZ,0172
011D  2047    JR   NZ,0166
011F  2020    JR   NZ,0141
```

PX　マクロ・コマンドのキャンセル

機能

マクロ・コマンドをキャンセルする.

書式

PX

解説

P コマンドで定義したマクロ・コマンドを取り消します.

例

```
-PX⏎
```

Q　デバッガの終了

機能

デバッガを終了する.

書式

Q

解説

デバッガを終了し，MSX-DOS2 のコマンド入力待ちに戻ります. MSX-S BUG1 のコントロール-C に相当し，MSX-S BUG2 より新しく追加されたコマンドです.

G コマンドでプログラムを実行させ，ユーザープログラムが"JP　0"などにより正常終了した場合も MSX-DOS2 のコマンド入力待ちに戻ることができます.

例

-q⏎

R　ディスク・ファイルの読み込み

機能

ディスク・ファイルを入力する.

書式

1)　R

2)　R<オフセット>

解説

Rコマンドは E コマンドと共に，ディスク・ファイルを読み込むために使用します. E コマンドで読み込むファイル名を指定し，R コマンドでそのファイルをディスクからメモリにロードします.

ファイル名の拡張子が，“.HEX”のときは，そのファイルがインテルフォーマットのヘキサファイルであるとして，バイナリに変換しながらメモリにロードし，他の拡張子のファイルはアブソリュートバイナリとして 0100H 番地から直接ロードします.

書式 1)はオフセットなしで(すなわち，“.HEX”のときは指定のアドレスに，それ以外では 0100H 番地から)ファイルをロードします. 書式 2)では指定されたオフセットを付加してロードします. “.HEX”以外のファイルでは，0100H＋<オフセット>がロード開始番地となります. R コマンドの実行中にエラー（指定のファイルが存在しない，チェックサムが正しくない，あるいはロード番地がシステムエリアとぶつかるなど）が生じるとクエスチョンマーク（?）を表示してロードを中止します. また，正しくロードできたときは以下のように，最後のロード番地＋1 を表示します.

例

```
-r⏎
Next  = XXXX    ⇨ロードしたモジュールのうち最も大きなもののエンドアドレス＋1
NextM = YYYY    ⇨最後にロードしたモジュールのエンドアドレス＋1
```

S　メモリへの書き込み

機能

メモリの内容を書き換えます．

書式

1）S

2）S<開始アドレス>

解説

開始アドレスが省略されたときはSコマンド用のデフォルトレジスタが利用されます．MSX-S BUG2 はこのコマンドを受け付けるとそのアドレスを表示し，続いてそのアドレスのメモリの内容を 16 進数で表示し入力待ちになります．適当な値が入力されますと，そのアドレスに入力されたデータが書き込まれ，次のアドレスに進みます．入力として許されるのは，

(1)結果が 0～255 の範囲の式．その値は表示されているアドレスに書き込まれ，アドレスが 1 つ増加されます．

(2)‘ ’（シングルクォート）か “ ”（ダブルクォート）で囲まれた任意長の文字列．
現在のアドレスから始まる連続したメモリ領域にその文字列が書き込まれます．漢字モードの使用時は漢字の入力も可能です．

(3)任意個の(1)と(2)の組み合せ．メモリ中に順番に書き込まれます．

(4) “- ⏎”．マイナス記号はメモリに書き込まれず表示アドレスを 1 つ前に戻します．

(5) “ ⏎” のみ．そのアドレスのメモリは変更されずアドレスが 1 つ増加されて次の行を表示します．

(6) “. ⏎”．ピリオドによって MSX-S BUG2 のコマンド入力待ちに戻ります．

例

```
-s 100⏎
0100 3E '>'    23⏎
0101 3F '?'    3F⏎
0102 D3 'ﾓ'    ⏎
0103 E1 '      8B⏎
0104 3D '='    55⏎
0105 D3 'ﾓ'    . ⏎
-s 100⏎
0100 23 '#'    'A'⏎
0101 3F '?'    "BCDE"⏎
0105 01 ^A     .  ⏎
```

漢字モード

```
-s 100⏎
0100 41 'A'    "春夏秋冬"⏎
0108 70 'p'    . ⏎
```

T　トレース（レジスタ表示あり）

機能

プログラムをトレースする.（レジスタ表示あり）

書式

1）T

2）T<式>

解説

書式1)では，ユーザープログラムを1命令だけトレースします．書式2)は<式>個の命令をトレースします．1命令トレースするたびにユーザーレジスタをXコマンドとおなじ書式で表示します．なお，ROM上のプログラムのトレースはできません．

<式>には式の他に数値を与えることもできますが，数値の後に何もつけていないもの，"H"のついたものは16進数とみなされます．10進数で表す場合は数値の後に"."，2進数で表す場合は数値の後に"!"をつけます．式については，第4章4.3式，定数，文字列を参照して下さい．

コンソールから任意のキーを入力するとトレースが中止され，MSX-S BUG2のコマンド入力待ちになります．

また，1命令トレースする度にユーザーレジスタの他にニーモニックを表示しますが，これはすでに実行してしまった命令ではなく，これから（次の"T"によって）実行される命令であることに注意して下さい．

例

漢字モード 0

```
-t 2⏎
P _ _0   A =A1 BC=0000 DE=0000 HL=A168
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=00FE P=0108 CALL 010D

P _ _0   A =A1 BC=0000 DE=0000 HL=A168
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=00FC P=010D  PUSH AF
```

TN トレース（レジスタ表示なし）

機能

プログラムをトレースする．（レジスタ表示なし）

書式

1） TN

2） TN<式>

解説

TN コマンドは，最後の命令が実行されるまでユーザーレジスタが表示されないことを除けばＴコマンドと全く同じです．

例

漢字モード 0

```
-tn 2↵
M _H_ONC A =03 BC=0002 DE=0331 HL=0000
P _ _0   A'=00 B'=0000 D'=0000 HL=0000
X=0000 Y=0000 I=00
S=0100 P=011E  ADD  A,30
```

V　メモリ内容の比較

機能

メモリ内容を比較する．

書式

V<ソース・アドレス>_<ソース・エンド>_<ディスティネーション・アドレス>

解説

これは＜ソース・アドレス＞から＜ソース・エンド＞までのメモリブロックの内容を＜ディスティネーション・アドレス＞からのメモリの内容と比較し，一致していない箇所のソース・アドレス，ディスティネーション・アドレスとそれぞれのデータを表示します．

例

漢字モード 0

```
-d 100 140
0100  3E 3F D3 E1 3D D3 34 CD -省略-
0108  00 02 C8 21 00 30 11 00
0110  40 3E 02 67 10 02 C3 00
0118  01 01 5A 2B ED E8 21 00
0120  3E 3F D3 E1 3D D3 E0 CD
0128  00 02 C8 21 00 30 11 00
0130  40 3E 02 CD 10 02 C3 00
0138  01 01 5A 2B ED B8 21 00
0140  CC
-v 100 120 120
0106 34   E0 0126
0113 67   CD 0133
0120 3E   CC 0140
```

W　ディスク・ファイルへの書き込み

機能

ファイルの書き込み／作成をする．

書式

W<開始アドレス>_<終了アドレス>

解説

Wコマンドはメモリの一部をディスクに書き込むために使います．ファイル名は前もってEコマンドで指定しておく必要があります．

これにより開始アドレスから終了アドレスまでが，Eコマンドで指定したファイル名で，ディスクに書き込まれます．そのファイルが既にディスクに存在しているときは，そのファイルの以前の内容は失われます．

例

```
-e ¥sample¥test.com⏎
-w 100 200⏎
```

X レジスタ内容の表示

機能

レジスタの内容を表示，変更する.

書式

X

X<レジスタ名>

解説

ユーザーレジスタの値は，いつでも書式1)のようにコマンドを入力すると表示することができます.

ANK モード

```
-x⏎
MZ_H_ENC A =00 BC=0000 DE=0000 HL=0000 S=0100 P=0100 RST    38
P _ _0   A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00
```

漢字モード 0

```
-x⏎
MZ_H_ENC A =00 BC=0000 DE=0000 HL=0000
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=0100 P=0100  RST  38
```

文字列 "MZ_H_ENC" および "P _ _O" はフラグの状態を示しています．前者は全てのフラグがオンのときの表示，後者は全てオフのときの表示です．"_"（アンダースコア）は定義されていないフラグビットを表します．詳細を以下に説明します．

M, P　サインフラグ．演算結果のMSBが1なら "M"（マイナス），0なら "P"（プラス）を意味します．

Z　ゼロフラグ．演算の結果が0ならZフラグは1になりコンソール上では"Z"と表示されます．

H　ハーフキャリーフラグ．H=1(Xコマンドでは "H" と表示される状態)になるのは加算の結果，アキュムレータのビット-4へキャリーがあがったとき，または減算の際アキュムレータのビット-4からボローが出たときです．

E, O　パリティまたはオーバーフローフラグ．算術演算のときにオーバーフローがおきると1になり，また論理演算のときの結果のパリティがイーブンのときに1となります．それ以外のときは0となります．
このフラグが1のときは "E" と表示され，0の時は "0" と表示されます．

N　　加／減算フラグ．最後に行われた演算が減算のとき１となり“N”と表示されます．

C　　キャリーフラグ．演算でキャリーが出ると１になり“C”と表示されます．

1行目の始めにフラグが表示されつぎにAレジスタ，続いてBC，DE，HLレジスタペア，そしてスタックポインタとプログラムカウンタ，さらにプログラムカウンタの指すメモリの内容が逆アセンブルされ表示されます．2行目にはF'，A'，BC'，DE'，HL'，さらにIX，IYとI(インタラプトページ・レジスタ)が順に表示されます．

レジスタの値を変更するときには書式2)のようにXの後に変更したいレジスタ名を書きます．レジスタ名として許されるのは以下のものです．

F，F'	フラグ
A，A'	アキュムレータ
B，B'	B，B'レジスタ
C，C'	C，C'レジスタ
D，D'	D，D'レジスタ
E，E'	E，E'レジスタ
H，H'	H，H'レジスタ
L，L'	L，L'レジスタ
BC，BC'	BC，BC'レジスタペア
DE，DE'	DE，DE'レジスタペア
HL，HL'	HL，HL'レジスタペア
X，IX	IXレジスタ
Y，IY	IYレジスタ
S，SP	スタックポインタ
P，PC	プログラムカウンタ
I	インターラプト・ページ・レジスタ
R	リフレッシュレジスタ

-X<レジスタ名>を入力するとMSX-S BUG2はそのレジスタの値を表示し，入力待ちになります．変更する場合は数字を入れて“⏎”を入力し，変更しない場合は“⏎”のみを入力します．

MSX-S BUG2起動時にユーザーレジスタの値はPCとSPが0100H，それ以外のレジスタは全て0000Hに初期化されます．

例

ANK モード

```
-x
MZ_H_ENC A =00 BC=0000 DE=0000 HL=0000 S=0100 P=0100  RET  38
P _ _0   A'=00 B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00
-x a
A =00   34
-x pc
PC=0100          1234
-x
MZ_H_ENC A =34  BC=0000 DE=0000 HL=0000 S=0100 P=1234  CALL 0005
P _ _0   A'=00  B'=0000 D'=0000 H'=0000 X=0000 Y=0000 I=00
```

漢字モード 0

```
-x
MZ_H_ENC A =00 BC=0000 DE=0000 HL=0000
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=0100 P=0100  RST  38
-x a
A =00   34
-x pc
PC=0100          1234
-x
MZ_H_ENC A =34  BC=0000 DE=0000 HL=0000
P _ _0   A'=00  B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=0100 P=1234  CALL 0005
```

Y　シンボルの表示

機能

シンボルを表示する．

書式

1)　Y

2)　Y<シンボル>

解説

書式 1) では登録されているシンボルとその値をすべて表示します．

書式 2) では登録されているシンボルのうち<シンボル>にマッチするシンボルとその値をすべて表示します．このときワイルドカード文字として，“?”と“*”が使えます．“?”はその位置にある文字が何であってもマッチすることを意味し，“*”は，それ以降許されるシンボルの最大長だけ“?”を入力したものとして扱われます．

例

```
-y du*⏎
0200 DUMMY
```

YR　シンボル・ファイルの読み込み

機能

シンボルファイルを読み込む.

書式

YR<ファイル名［.拡張子］>

解説

シンボルファイルをディスクからロードして，現在登録されているシンボルに追加するために使います．ファイル名の拡張子が省略された場合には，拡張子として，".SYM"が入力されたものとして扱われます．

例

```
-yr ¥sample¥test.sym⏎
```

YS　シンボルの定義

機能

シンボルを定義する.

書式

YS<シンボル>

解説

新たにシンボルを定義するのに使います.

```
-ys dummy⏎
```

と入力すると,

```
DUMMY  =
```

と表示されますので, 適当な数値を入力します. 以後“DUMMY”は今入力した値と等価なものとして扱われます.

例

```
-ys dummy⏎
DUMMY  = 200⏎
```

YX　シンボルの消去

機能

シンボルを削除する.

書式

YX [<シンボル>]

解説

既に登録してあるシンボルを消去するために使います.

```
-yx⏎
```

と入力すると現在登録されているシンボルがすべてクリアされます. YX の後にシンボルを書けば, そのシンボルのみがクリアされます. ワイルドカード文字を使用した場合は, マッチする全てのシンボルがクリアされます.

例

```
-yx dummy⏎
```

付録 A　プログラム開発におけるデバッグ作業とその実際

この付録は，アセンブラによるプログラムを実際に開発する過程で，画面に表示されたものをそのままの形で活字にしたものです．何かと独り言の多いプログラマですが，どうか最後までつき合って，デバッグの実践の感覚を摑んで下さい．なお，このとおりに実行するには，MSX-S BUG2 の他に，MSX-DOS2 TOOLS に付属のいくつかのプログラムが必要です．

test.mac は，アセンブラの練習のために書いたプログラムです．"Hello!!"というメッセージを1行画面に表示するだけのプログラムです．

スクリーンは漢字モード 0（MODE＝80），ソース・ファイルは，カレントドライブの¥sample にあり，KID（MSX-DOS2 TOOLS 付属の漢字スクリーン・エディタ）で作成しました．

```
A>type ¥sample¥test.mac
        .z80
start::
        ld      hl,(6)
        ld      sp,hl
        ld      hl,tstmsg
        call    puts
        jp      0
puts::
        push    hl
        ld      a,(hl)
        or      a
        ret     z
        ld      e,a
        ld      c,2
        call    5
        pop     hl
        inc     hl
        jr      puts

        dseg
tstmsg::
        defb    "Hello!!",0dh,0ah,0

        cseg

        end     start
```

ソース・ファイルはちゃんとできているかしら？

```
A>m80 =¥sample¥test

No Fatal error(s)
```

ふむふむ...
それでは，いざアセンブル開始！
へっへっ，完璧．

```
A>l80 ¥sample¥test,¥sample¥test/n/y/c

MSX.L-80  2.00  01-Mar-89  (c) 1989 Microsoft
```

お次はリンクでマシン語ファイルの完成ね．
念のため/Y でシンボル・ファイルも作っとこっ＊＊＊

```
Data    0103    0128    <   37>

49149 Bytes Free
[010D   0128         1]
```

OK!

```
A>¥sample¥test↵
```

よし，動かしてみるぞ...せーのっと...
へっ..

```
Hello!!
Hello!!
Hello!!
```

うっ...

```
Hello!!
Hello!!
Hello!!
Hello!!
Hello!!
```

あれーっ

```
Hello!!
Hello!!
Hell
```

（あわててコントロール-Cを押す）

```
*** Ctrl-Cが押されました
```

ソースを点検しましたが，どうしてこんなことになったのかわかりません．MSX-S BUG2で，デバッグするしかないでしょう．（シンボル・ファイルを作っておいてよかった...）

```
A>sbug2m ¥sample¥test↵
Can not load to mapper ram
A>ramdisk↵
RAMDISK=160K
```

sbug2m.comをロードできるだけの容量がなかったわけね．（容量全部RAMディスクに割り当てているってこと）マッパーRAMを少し解放して...

sbug2m.comは1セグメント必要だからRAMディスクは160K-16Kの144Kに設定っと．（RAMディスクに入っている大切なファイルを消さないように気を付けてね）

```
A>ramdisk 144↵
RAMディスク上の全てのデータを消去しますか(Y/N)?y
A>sbug2m ¥sample¥test↵
MSX Symbolic Instruction Debugger 1.00
           Mapper Version
  (C)  BUG / ASCII CORPORATION  1989

Next  = 0180
NextM = 0180
Symbol
```

まずは，逆アセンブル．
....COMファイルだからねっ...

```
-l↵
0100  C30D01    JP    010D [START]
TSTMSG:
0103  48        LD    C,B
0104  65        LD    H,L
0105  6C        LD    L,H
0106  6C        LD    L,H
0107  6F        LD    L,A
0108  21210D    LD    HL,0D21
010B  0A        LD    A,(BC)
010C  00        NOP
```

フムフム，このあたりはメッセージのデータみたいだけど...ソースでは最後に書いたハズなのに...
... あっそうだっ！L-80でリンクするとDSEG,CSEGの順に置かれるんだっけ．（とわざとらしく思いだして，バグの巧妙さを強調する伏線を張っておく）

```
START:
010D  2A0600    LD    HL,(0006)
```

うひゃー，ソースとおんなじ（あたりまえか）!!

```
-l↵
0110  F9        LD    SP,HL
0111  210301    LD    HL,0103 [TSTMSG]
0114  CD1A01    CALL  011A [PUTS]
0117  C30000    JP    0000
PUTS:
011A  E5        PUSH  HL
011B  7E        LD    A,(HL)
```

```
011C  B7        OR    A,A
011D  C8        RET   Z
011E  5F        LD    E,A
011F  0E02      LD    C,02
-g/puts⏎

PUTS:
P _ _0   A =00 BC=0000 DE=0000 HL=0103
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A  PUSH HL
-g/puts⏎
H
PUTS:
PZ_ _E   A =00 BC=0002 DE=0048 HL=0104
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A  PUSH HL
-g/puts⏎
e
PUTS:
PZ_ _E   A =00 BC=0002 DE=0065 HL=0105
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A  PUSH HL
-xp⏎
P =011A         100
-g/puts⏎

PUTS:
P _ _E   A =00 BC=0002 DE=0065 HL=0103
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A PUSH HL
-g/puts(900)⏎
Hello!!
Hello!!
Hello!!
Hello!!
Hello!!
Hel

Ctrl-Cが押されました
-xp⏎
P =011A         100
-g/puts(9)⏎
Hello!!

PUTS:
PZ_ _E   A =00 BC=0002 DE=000A HL=010C
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A  PUSH HL
-t⏎
PZ_ _E   A =00 BC=0002 DE=000A HL=010C
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E7 P=011B  LD   A,(HL)
-t⏎
PZ_ _E   A =00 BC=0002 DE=000A HL=010C
P _ _0   A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E7 P=011C  OR   A,A
```

こうすれば，サブルーチンの直前で止まるハズっと．それっ...

もう１回．
ほぉーっ，最初の１文字．

調子にのってもう１回．
２文字目！

プログラム・カウンタをセットして...
もう１回最初（100H 番地）から実行．

ええいっ，面倒だ，９文字１ぺんにっ！
（改行があるのを忘れずにねっ！）
あっ！'9'を"900"って入力しちゃった．

ここで慌てず CTRL-C.

もう１度プログラム・カウンタをセット．
今度は落ち着いて...

ここからは，小出しに実行．

```
-t⏎
PZ_ _E    A =00 BC=0002 DE=000A HL=010C
P _ _O    A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E7 P=011D  RET  Z
```

Aレジスタは0だから...

```
-t⏎
PZ_ _E    A =00 BC=0002 DE=000A HL=010C
P _ _O    A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=010C  NOP
```

これでサブルーチンを抜けるハズ！！
あれっ？"JP 0"のハズでしょ.

```
-l$⏎
010C  00         NOP
START:
010D  2A0600     LD     HL,(0006)
0110  F9         LD     SP,HL
0111  210301     LD     HL,0103 [TSTMSG]
0114  CD1A01     CALL   011A [PUTS]
0117  C30000     JP     0000
PUTS:
011A  E5         PUSH   HL
011B  7E         LD     A,(HL)
011C  B7         OR     A,A
```

私は，いったい何処にいるの？
おやおや，ここはメッセージの最後だわ.
これじゃ，もう1回スタートしちゃうわけだ.

このPUSH命令の位置がバグの原因みたい.

```
-l⏎
011D  C8         RET    Z
011E  5F         LD     E,A
011F  0E02       LD     C,02
0121  CD0500     CALL   0005
0124  E1         POP    HL
0125  23         INC    HL
0126  18F2       JR     011A [PUTS]
0128  19         ADD    HL,DE
0129  B4         OR     A,H
012B  018600     LD     BC,0086
```

RET命令は最後にPUSHした番地に帰るんだから....
...おっ恐ろしい. たまたま，START：の直前に帰ったからよかったものの. 危うくシステムダウンにつながる，きわどいバグでした.

```
-aputs⏎
PUTS:
011A  E5         PUSH  HL
                 LD A,(HL)
011B  7E         LD    A,(HL)
                 OR A
011C  B7         OR    A,A
                 RET Z
011D  C8         RET   Z
                 PUSH HL
011E  5F         LD    E,A
                 .
```

それでは，こんなふうにしたら...

```
-xp⏎
P =010C          100
-g/puts⏎
```

これで，もう1回試してみよう.

```
PUTS:
PZ_ _E    A =00 BC=0002 DE=000A HL=0103
P _ _O    A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0E9 P=011A  LE   A,(HL)
-g/#⏎
Hello!!
PZ_ _E    A =00 BC=0002 DE=000A HL=010C
P _ _O    A'=00 B'=0000 D'=0000 H'=0000
X=0000 Y=0000 I=00
S=D0EB P=0117  JP    0000
-q⏎
```

いまのスタック・トップの番地まで戻って来るかな？

... OK!OK!

```
A>kid ¥sample¥test.mac
・・・
```

それじゃ，ソースをちゃんと直しましょ．

付録B　コマンド一覧

コマンド名	用　　途	ページ
?	コマンド一覧の表示	27
A	アセンブル	28
C	コール命令のルーチンをパスするトレース(レジスタ表示あり)	29
CN	コール命令のルーチンをパスするトレース(レジスタ表示なし)	31
D	メモリ・ダンプ	32
E	ファイル名の指定	34
F	メモリの初期化	36
G	プログラムの実行	37
H	演算機能	39
I	入出力ポートからの読み込み	40
IO	入出力ポートへの書き込み	41
K	パーマネント・ブレイクポイントの設定	42
KX	パーマネント・ブレイクポイントの解除	43
L	逆アセンブル	44
M	メモリ上のデータの転送	45
N	メモリ・サーチ	46
O	入出力ポートへのデータ出力	47
P	マクロ・コマンドの定義及び実行	48
PX	マクロ・コマンドのキャンセル	49
Q	デバッガの終了	50
R	ディスク・ファイルの読み込み	51
S	メモリへの書き込み	52
T	トレース(レジスタ表示あり)	53
TN	トレース(レジスタ表示なし)	54
V	メモリ内容の比較	55
W	メモリのディスクへのセーブ	56
X	レジスタ内容の表示	57
Y	シンボルの表示	60
YR	シンボルファイルの読み込み	61
YS	シンボルの定義	62
YX	シンボルの削除	63

付録C Command summary

?	命令一覧の表示
A [番地]	ニーモニック入力
C[N] [回数値]	CALL 命令以外のトレース
D [開始番地] [終了番地]	メモリー内容の表示
E ファイル名 [ファイル名]	ファイル名のセット
F 開始番地 終了番地 データ	データによる初期化
G [開始番地] [/[番地[(回数値)]]...]	プログラムの実行
H 式 [式]	数値計算
I [ポート]	ポートからの入力
IO [ポート]	ポートに対する入力/出力
K 番地[(回数値)] [番地[(回数値)]]...	ブレイクポイントの設定
K	ブレイクポイントの表示
KX [番地] ...	ブレイクポイントの消去
L [開始番地] [終了番地]	アセンブルリストの表示
M 開始番地 終了番地 目的番地	メモリー間の転送
N 開始番地 終了番地 データ [データ]...	データの検索
O [ポート] データ	ポートへのデータ出力
P	マクロ命令の実行
P 命令[:命令]...	マクロ命令の設定
PX	マクロ命令の消去
Q	デバッガの終了
R [オフセット]	ファイルの読み込み
S [番地]	メモリー内容の書き換え
T[N] [回数値]	プログラムのトレース
V 開始番地 終了番地 目的番地	メモリーの内容比較
W 開始番地 終了番地	ファイルの書き込み/作成
X	レジスタ内容の表示
Xr (r = A,B,C,D,E,H,L,BC, etc.)	レジスタ内容の書き換え
Y [シンボル名]	シンボル名と番地の表示
YR ファイル名[.TYPE]	シンボルファイルの読み込み
YS シンボル名	シンボルの定義
YX [シンボル名]	シンボルの削除

お問い合わせについて

弊社では厳重に梱包した上，細心の注意を払って製品を発送しております．万一，輸送上のトラブルが起こった場合にはご一報いただければ新しいものと交換いたします．

マニュアル作成にあたり，なるべく詳細な説明をするように心がけたつもりですが，理解できないところは実際にコンピュータと向きあって納得のいくまで確かめて下さい．また，他のページを参照するのもひとつの方法です．それでも疑問点が解決できないときは，購入された販売店に問い合わせるか，㈱アスキー　ユーザーサポート（直通電話 03-498-0299）までお電話いただければ，係がお答えします．しかしながら，回線が混み合いご迷惑をかけることもありますので，なるべくお手紙にてお願いいたします．その際には，下記の要領で記入して下さい．記入されてない項目が一つでもありますと，回答できかねる場合があります．十分注意して下さい．

また，本製品以外に対してのご意見，ご希望がございましたら，弊社までおよせください．

記

1.送付先　〒107-24　東京都港区南青山 6-11-1　スリーエフ南青山ビル
株式会社アスキー　ユーザーサポート係
TEL　03-498-0299
（祝祭日を除く月～金曜日，10：00～12：00，13：00～17：00）

2.必要項目

(1)お客様の氏名，住所（郵便番号），電話番号（市外局番も含む）

(2)製品名，製品シリアル番号

(3)機器構成

本体装置名，メモリバイト数

CRT 装置名，フロッピーディスク装置名

プリンタ装置名

その他 I/O，I/F 装置名

(4)お問い合わせ内容

お問い合わせの内容は，できるだけ製品のマニュアルに記述されている用語を用いて，具体的かつ明確に記述してください．なお，障害と思われる現象については，その現象を再現可能な情報が必要です．当社で再現できないものは，調査ができません．その現象が発生するまでの操作手順，データを必ず添付して下さい．データディスクがある場合は，そのコピーも同封していただくと調査がスピーディーになります．

また，お客様固有と思われるアプリケーションの設計，作成，運用，保守については，当社のサポート範囲外ですので，お問い合わせいただいても回答できません．ご了承下さいますようお願いいたします．

MSX-S BUG2 USER'S MANEAL

1989年4月1日　第1版第1刷発行

監修／編集　株式会社アスキー　システム機器事業部

発 行 所　株式会社アスキー
〒107-24　東京都港区南青山6-11-1　スリーエフ南青山ビル

振　　替　東京 4-161144
TEL.　(03) 486-7111

正 誤 表

33ページ 誤

例

ANK モード

```
-d 100 130
0100    44 49 52 0D 0A 0D 0A 5B 8B 40 94 5C 5D 20 20 83 DIR____[ぉ@え¥]  ◆
0110    66 83 42 83 58 83 4E 8F E3 82 CC 83 74 83 40 83 f◆B◆X◆Nって※フ◆t◆@◆
0120    43 83 8B 96 BC 82 F0 95 5C 8E A6 82 B7 82 E9 2E C◆ぉカシ※みお¥ょう※キ※の.
0130    0D
```

(注意) ※は半角のクラブの代用で，平仮名，ダイヤは全て半角です．

漢字モード 0

```
-d 100 130
0100  44 49 52 0D 0A 0D 0A 5B DIR____[
0108  8B 40 94 5C 5D 20 20 83 機能]  デ
0110  66 83 42 83 58 83 4E 8F  ィスク上
0108  E3 82 CC 83 74 83 40 83  のファイ
0120  43 83 8B 96 BC 82 F0 95  ル名を表
0128  5C 8E A6 82 B7 82 E9 2E  示する.
0130  0D                       _
```

⇩

正

例

ANK モード

```
-d 100 130
0100    44 49 52 0D 0A 0D 0A 5B 8B 40 94 5C 5D 20 20 83 DIR____[ぉ@え¥]  ◆
0110    66 83 42 83 58 83 4E 8F E3 82 CC 83 74 83 40 83 f◆B◆X◆Nって♣フ◆t◆@◆
0120    43 83 8B 96 BC 82 F0 95 5C 8E A6 82 B7 82 E9 2E C◆ぉカシ♣みお¥ょう♣キ♣の.
0130    0D                                              _
```

漢字モード 0

```
-d 100 130
0100  44 49 52 0D 0A 0D 0A 5B DIR____[
0108  8B 40 94 5C 5D 20 20 83 機能]  デ
0110  66 83 42 83 58 83 4E 8F  ィスク上
0108  E3 82 CC 83 74 83 40 83  のファイ
0120  43 83 8B 96 BC 82 F0 95  ル名を表
0128  5C 8E A6 82 B7 82 E9 2E  示する.
0130  0D                       _
```

「MSX-S BUG2」ユーザーズマニュアル正誤表その2

1989年4月14日

頁	行	誤	正
P.17	22	MSX Symbolic Instruction Debugger 1.00	MSX Symbolic Instruction Debugger 2.10
P.65	26	MSX Symbolic Instruction Debugger 1.00	MSX Symbolic Instruction Debugger 2.10